AI Engineer Intern Task

Company: NoBrokerage.com

Objective: Build an intelligent, GPT-like **chat interface** that understands user queries like

> "3BHK flat in Pune under ₹1.2 Cr"
>
> and responds with:

1. A short, helpful **summary** (generated from the database only).
2. A list of **relevant property or project cards** from a provided CSV file.

CSV File

project.csv 10.6 KB

ProjectAddress.csv 4.6 KB

ProjectConfiguration.csv 4.4 KB

ProjectConfigurationVariant.csv 29.9 KB

🎯 Project Goal

You'll build a **Chat + Search system** that helps users discover properties and projects through **natural language** instead of filters.

We'll provide you with a **CSV file** containing project data (with fields like name, city, price, bhk, possession, etc.).

Your AI chatbot should:

- Understand what the user is asking.
- Extract key filters like **city, BHK, budget, project name, and locality**.
- Search the CSV data for relevant matches.
- Generate a 4-5 **sentence response summary** (only using CSV data).
- Return a **list of property/project cards** with basic details.

🧩 Core Functionalities to Implement

1. Chat Interface (Frontend)

- Build a **simple chat UI** (React or Next.js preferred or stramlit).
- UI can be similar like ChatGPT

---

2. Natural Language Query Understanding (Backend)

Your backend should:

- Parse user messages and extract filters like:
  - City (e.g., Pune, Mumbai)
  - BHK (e.g., 2BHK, 3BHK)
  - Budget (e.g., under ₹1.2 Cr)
  - Readiness (Ready-to-move / Under Construction)
  - Locality or soft intents (e.g., near metro, near IT park)
  - Project Name (optional)
- You can use **regex**, **rule-based parsing**, or **small open-source LLMs** (if available).

---

3. Search & Retrieval (Data Layer)

- Load the provided CSV file into any local store:
  - PostgreSQL / JSON / Pandas DataFrame – your choice.
- Retrieve results by applying parsed filters.
- Optional (Bonus):

  Add **semantic search** using embeddings (e.g., Sentence Transformers, `pgvector`, or OpenSearch).

---

4. Summarization Logic

- After fetching results:
  - Generate a **short summary (2–4 lines)** describing the best-matched properties.
  - Example:

    > "Within ₹1.2 Cr, most 3BHK ready homes in Pune are found near Wakad and Baner. 6 listings have metro access and club amenities."

- Summaries must be **grounded only in CSV data**.

  No hallucination or external info.
- If no results found — return a graceful fallback:

  > "No ready 3BHK options found under ₹1.2 Cr in Baner. Expanding search to Wakad and Thergaon found 4 options."

---

5. Property/Project Cards

Each result should include:

- Title
- City + Locality
- BHK
- Price (formatted as ₹X Cr / ₹X L)
- Project Name
- Possession Status (Ready / Under Construction)
- Top 2–3 amenities
- CTA URL (`/project/<slug>`)

⚙️ Recommended Tech Stack

Layer	Recommended Tools
Frontend	React.js / Next.js
Backend	Node.js / Express or Next.js Route Handlers
Database	PostgreSQL / Local JSON
AI / NLP	OpenAI / Hugging Face models / Regex-based parsing
Optional	pgvector, sentence-transformers, OpenSearch
Language	JavaScript / TypeScript
Version Control	GitHub Repository

🧠 What You'll Be Evaluated On

Criteria	Weight	Description
Query Understanding	30%	Correct extraction of filters from user query
Result Accuracy	25%	Relevance of search results from CSV
Summary Generation	20%	Quality, clarity, and grounding of summary
Code Quality	15%	Clean structure, readability, documentation
UI/UX Polish	10%	Simple, user-friendly chat experience

🧰 Deliverables

1. GitHub repository with:
   - Code (frontend + backend)
   - `/data/` folder (with a sample CSV)
   - `README.md` (setup guide + examples)
   - `.env.example` file (if using API keys)
2. Live demo link.
3. Live Deployment link (optional bonus).
4. Github username to share → Prathameshzad and https://github.com/batty-sk

⏰ Timeline & Deadline

- **Deadline: Within 3 days**
- **Submission Format:** GitHub Repo URL + short Loom video demo (optional)

🚫 Important Rules

- Do **not** hardcode responses. All data must come from CSV.
- Do **not** scrape or use external listing APIs.
- Summaries must be factual — no made-up properties.
- Keep the interface minimal and intuitive.